EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

2. Authorization for this examiner's amendment was given in a telephone interview with Jeffrey Hsu on 12/15/2009.

3. Please replace all prior versions or listings of the claims with the following:

1. (Canceled)

2. (Currently Amended) The method of claim [[1]]11, wherein each timeslice is between 10 and 100 microseconds.

3. (Currently Amended) The method of claim [[1]]11, wherein the array of threads requesting immediate CPU resource allocation includes a first-in-first-out (FIFO) structure.

4. (Canceled)

5. (Currently Amended) The method of claim [[1]]11, wherein the step of suspending the currently executing thread includes:

receiving a self-suspend request from the currently executing thread.

6. (Previously Presented) The method of claim 5, further comprising:

advancing the index pointer by one time slot;

removing a list of any threads to be executed at the indexed time slot and appending them to the array of threads requesting immediate CPU resource allocation; determining whether the array of threads requesting immediate CPU resource allocation is empty;

returning to the step of advancing the index pointer by one slot if it is determined that the array of threads requesting immediate CPU resource allocation does not contain any threads; and

removing and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the array of threads requesting immediate CPU resource allocation is not empty.

7. (Currently Amended) The method of claim [[1]]11, further comprising:

receiving an external event interrupt requesting CPU resource allocation for a new thread;

calculating a next time slot during which the currently executing thread should next resume execution;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation;

determining a type of thread currently executing;

activating the new thread if no thread is currently executing;

performing the following steps if it is determined that a non-idle thread is currently executing:

suspending the currently executing thread;

appending the currently executing thread to the end of the array of threads requesting immediate CPU resource allocation; and

activating the new thread;

performing the following steps if it is determined that an idle thread is currently executing:

suspending the currently executing thread; and

activating the new thread.

8. (Currently Amended) A method for scheduling thread execution, comprising:

maintaining a plurality of circular array structures associated with a plurality of discrete thread priorities, each having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;

configuring each time slot in each of the circular arrays to include a queue of threads scheduled for execution during that time slot;

maintaining at least one pointer index for referencing one time slot in each of the circular arrays, whereby advancement through the circular arrays is provided by advancing the pointer index;

maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays;

assigning each thread to be executed a specific priority;

incrementing the index pointer by one slot;

removing, for each of the plurality of circular arrays, each queue of threads for the indexed time slot;

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array;

determining whether the array of threads requesting immediate CPU resource allocation associated with a first circular array contains any threads;

proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty;

extracting [[its]]a top thread if the array of threads requesting immediate CPU resource allocation contains any threads;

determining whether a priority of the top thread is greater than a priority of the currently executing thread;

calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread; performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread:

suspending the currently executing thread;

activating the top thread; and

calculating the time of next execution for the suspended thread;

determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed; and

proceeding to the next array of threads requesting immediate CPU resource allocation if it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed.

9. (Original) The method of claim 8, wherein each of the plurality of circular arrays corresponds to one of four assigned priority levels: a non-real-time priority; a soft-real-time priority; a hard-real-time priority; and a critical-real-time priority.

10. (Original) The method of claim 8, further comprising:

receiving an external event interrupt requesting CPU resource allocation for a new thread;

calculating a next time slot during which the currently executing thread should next resume execution;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation;

determining whether a priority of the new thread is greater than a priority of the currently executing thread;

appending the new thread to the end of the array of threads requesting immediate CPU resources for the associated priority if it is determined that the priority of the new thread is not greater than the priority of the currently executing thread; and performing the following steps if it is determined that the priority of the new thread is greater than the priority of the currently executing thread:

suspending the currently executing thread;

calculating the time for next execution for the currently executing thread

appending the currently executing thread to array associated with the calculated time slot; and

activating the new thread.

11. (Currently Amended) A method for scheduling thread execution, comprising:

maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;

configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot;

maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index;

maintaining an array of threads requesting immediate CPU resource allocation based on the queue of threads;

calculating a next time slot during which a currently executing thread should next resume execution;

appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot;

identifying a next sequential non-empty time slot containing a queue of threads scheduled for execution during that time slot;

updating the pointer index to point to the identified next sequential non-empty time slot;

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;

removing the thread at ~~the~~ a top of the array of threads requesting immediate CPU resource allocation;

determining whether the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;

maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;

suspending [[a]]the currently executing thread; and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread.

12. (Canceled)

13. (Currently Amended) The computer-readable storage medium of claim [[12]]22, wherein each timeslice is between 10 and 100 microseconds.

14. (Currently Amended) The computer-readable storage medium of claim [[12]]22, wherein the array of threads requesting immediate CPU resource allocation includes a first-in-first-out (FIFO) structure.

15. (Canceled)

16. (Currently Amended) The computer-readable storage medium of claim [[12]]22, wherein suspending the currently executing thread further comprises receiving a self-suspend request from the currently executing thread.

17. (Previously Presented) The computer-readable storage medium of claim 16, further causing the computer to perform:

advancing the index pointer by one time slot;

removing a list of any threads to be executed at the indexed time slot and appending them to the array of threads requesting immediate CPU resource allocation;

determining whether the array of threads requesting immediate CPU resource allocation is empty;

returning to the step of advancing the index pointer by one slot if it is determined that the array of threads requesting immediate CPU resource allocation does not contain any threads; and

removing and activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the array of threads requesting immediate CPU resource allocation is not empty.

18. (Currently Amended) The computer-readable storage medium of claim [[12]]22, further causing the computer to perform:

receiving an external event interrupt requesting CPU resource allocation for a new thread;

calculating a next time slot during which the currently executing thread should next resume execution;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation;

determining the type of thread currently executing;

activating the new thread if no thread is currently executing;

performing the following steps if it is determined that a non-idle thread is currently executing:

suspending the currently executing thread;

appending the currently executing thread to the end of the array of threads requesting immediate CPU resource allocation; and

activating the new thread;

performing the following steps if it is determined that an idle thread is currently executing:

suspending the currently executing thread; and

activating the new thread.

19. (Currently Amended) A computer-readable storage medium storing a program for scheduling thread execution, the program, when executed by a processor, causing a computer to perform:

maintaining a plurality of circular array structures associated with a plurality of discrete thread priorities, each having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;

configuring each time slot in each of the circular arrays to include a queue of threads scheduled for execution during that time slot;

maintaining at least one pointer index for referencing one time slot in each of the circular arrays, whereby advancement through the circular arrays is provided by advancing the pointer index;

maintaining an array of threads requesting immediate CPU resource allocation for each of the plurality of circular arrays;

assigning each thread to be executed a specific priority;

incrementing the index pointer by one slot;

removing, for each of the plurality of circular arrays, each queue of threads for the indexed time slot;

appending each removed thread to the array of threads requesting immediate CPU resource allocation associated with its respective circular array;

determining whether the array of threads requesting immediate CPU resource allocation associated with a first circular array contains any threads;

proceeding to a next circular array if the array of threads requesting immediate CPU resource allocation is empty;

extracting [[its]]a top thread if the array of threads requesting immediate CPU resource allocation contains any threads;

determining whether a priority of the top thread is greater than a priority of the currently executing thread;

calculating a time for next execution of the top thread if it is determined that the priority of the top thread is not greater than the priority of the currently executing thread;

performing the following steps if it is determined that the priority of the top thread is greater than a priority of the currently executing thread:

suspending the currently executing thread;

activating the top thread; and

calculating the time of next execution for the previously executing thread;

determining whether each of the array of threads requesting immediate CPU resource allocation associated with each of the circular arrays has been processed; and

proceeding to the next array of threads requesting immediate CPU resource allocation if it is determined that not all arrays of threads requesting immediate CPU resource allocation have been processed.

20. (Previously Presented) The computer-readable storage medium of claim 19, wherein each of the plurality of circular arrays corresponds to one of four assigned priority levels: a non-real-time priority; a soft-real-time priority; a hard-real-time priority; and a critical-real-time priority.

21. (Previously Presented) The computer-readable storage medium of claim 19, the program further causing the computer to perform:

receiving an external event interrupt requesting CPU resource allocation for a new thread;

calculating a next time slot during which the currently executing thread should next resume execution;

determining whether the external event interrupt is requesting immediate CPU resource allocation;

appending the new thread to a queue on a time slot on the circular array if it is determined that the external event interrupt is not requesting immediate CPU resource allocation;

determining whether a priority of the new thread is greater than a priority of the currently executing thread;

appending the new thread to the end of the array of threads requesting immediate CPU resources for the associated priority if it is determined that the priority of the new thread is not greater than the priority of the currently executing thread; and

performing the following steps if it is determined that the priority of the new thread is greater than the priority of the currently executing thread:

suspending the currently executing thread;

calculating the time for next execution for the currently executing thread

appending the currently executing thread to array associated with the calculated time slot; and

activating the new thread.

22. (Currently Amended) A computer-readable storage medium storing a program for scheduling thread execution, the program<u>, when executed by a processor,</u> causing a computer to perform:

maintaining a circular array structure having a plurality of time slots therein, wherein each of the plurality of time slots corresponds to a timeslice during which CPU resources are allocated to a particular thread;

configuring each time slot in the circular array to include a queue of threads scheduled for execution during that time slot;

maintaining a pointer index for referencing one time slot in the circular array and whereby advancement through the circular array is provided by advancing the pointer index;

maintaining an array of threads requesting immediate CPU resource allocation;

calculating a next time slot during which a currently executing thread should next resume execution;

appending the currently executing thread to the queue of threads scheduled for execution at the calculated time slot;

identifying a next sequential non-empty time slot;

updating the pointer index to point to the identified next sequential non-empty time slot;

appending any contents of the indexed time slot to the array of threads requesting immediate CPU resource allocation;

removing the thread at ~~the~~ a top of the array of threads requesting immediate CPU resource allocation;

determining whether the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;

maintaining execution of the currently executing thread for the following time slot if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is identical to the currently executing thread;

suspending [[a]]<u>the</u> currently executing thread; and

activating the thread at the top of the array of threads requesting immediate CPU resource allocation if it is determined that the thread at the top of the array of threads requesting immediate CPU resource allocation is not identical to the currently executing thread.

Conclusion

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to CAROLINE ARCOS whose telephone number is (571)270-3151. The examiner can normally be reached on Monday-Thursday 7:00 AM to 5:30 PM.

5. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on 571-272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

6. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would

like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Meng-Ai An/
Supervisory Patent Examiner, Art Unit 2195

/Caroline Arcos/
Examiner, Art Unit 2195